SYSTEM AND METHOD FOR NETWORK SECURITY

Field of the Invention

The invention relates to the field of communications, and more particularly to advanced network security.

Background of the Invention

The consistent demand for computer and other network services has increased the need for better network security tools. A variety of techniques have been deployed to shield networks from hacking and other intrusions. Those protective techniques may be categorized as either risk avoidance systems or risk management systems.

Risk avoidance techniques involve introducing a barrier to prevent inappropriate entry into a network. Such systems place reliance on keeping intruders out of the network entirely, rather than monitoring inappropriate network traffic after logging in. Risk avoidance systems include dedicated network firewalls and mandatory encryption over the network. Commercial examples include Gauntlet™, Firewall-1™, Guardian™, BorderWare™ and others.

Risk management approaches, in contrast, adopt the philosophy that a network can not keep everyone out, and so rely upon detection of intrusive activity after logging in. Unfortunately, intrusion detector systems often lend a false sense of security to systems administrators, while not really solving the

underlying security problem. Intrusion detector systems produce a high rate of false positive identification, by inaccurately reporting legitimate network activity as suspicious. Intrusion detector systems also often overwhelm a systems administrator with too much detail about network behavior, and moreover are configured to trigger a report only after discovery of a network attack. Of course, at this point in time it is too late to prevent the attack or often to remedy much of the possible damage. Commercial examples include ISS RealSecure™, NetRanger™, TACAS+, NFR and others.

After-the-fact auditing systems provide another type of tool used under the risk management approach. Auditing systems are implemented as a host-based technique, in which a central server running the operating system logs the activity of client computers in a central storage area. However, the host computer running the audit system itself may be susceptible to being attacked internally or externally, creating a point of vulnerability in the overall surveillance.

Some other auditing products, such as Session Wall-3™ from AbirNet, employ so-called sniffer technology to monitor network traffic. Data streams collected by such products look for specific types of network traffic, for example, detecting electronic mail uploads by monitoring port 25 for simple mail transfer protocol (SMTP) events. However, most networks carry a large amount of traffic and sniffer type tools do not help sift through the volume. Other drawbacks exist.

More robust and comprehensive network security technology is desirable.

Summary of the Invention

The invention overcoming these and other problems in the art relates to a system and method for network security capable of comprehensive network surveillance. The invention incorporates both network monitoring ports and analysis tools which enable a systems administrator to unobtrusively, but thoroughly, profile the entire range of network activity. The invention is incorporated into computer and other installations at the network level, and generally includes a dedicated observation port which passes the entire range of network traffic into a system interpreter.

The collected information, typically in the form of packets, is subjected to a series of reductions to network sessions, metadata and eventually to statistical or other summary presentations. The invention thus subjects network traffic to a hierarchical series of real-time or forensic treatments, in which no type of data or network activity is excluded. Because the invention is only reading data at the network level and does not rely upon a central server running other tasks, the security protection offered is difficult or impossible to circumvent or corrupt. Because the entire data stream of the network is captured and profiled and profiling is not dependent on one subset of port

assignments or boundary conditions, forensic inspection of past network activity is enhanced.

Brief Description of the Drawings

The invention will be described with respect to the accompanying drawings, in which like elements are represented by like numbers.

Fig. 1 illustrates a network architecture for security according to the invention.

Fig. 2 is a flow chart illustrating surveillance and auditing processing according to the invention.

Fig. 3 illustrates a presentation interface for viewing and analyzing data collected by the invention.

Fig. 4 illustrates the operation of an interpreter module according to the invention.

Fig. 5 illustrates the operation of an assembler module and parser module according to the invention.

Detailed Description of Preferred Embodiments

The invention will be described with respect to a network architecture illustrated in Fig. 1, in which a network observation port 104 monitors a network data stream 144 traveling over a network 142. Network 142 may be or include as a segment any one or more of, for instance, the Internet, an intranet, a PAN (Personal Area Network), a LAN (Local Area Network), a WAN (Wide

Area Network) or a MAN (Metropolitan Area Network), a frame relay connection, an Advanced Intelligent Network (AIN) connection, a synchronous optical network (SONET) connection, a digital T1, T3 or E1 line, Digital Data Service (DDS) connection, DSL (Digital Subscriber Line) connection, an Ethernet connection, an ISDN (Integrated Services Digital Network) line, a dial-up port such as a V.90, V.34 or V.34bis analog modem connection, a cable modem, an ATM (Asynchronous Transfer Mode) connection, or FDDI (Fiber Distributed Data Networks) or CDDI (Copper Distributed Data Interface) connections.

Network 142 may furthermore be or include as a segment any one or more of a WAP (Wireless Application Protocol) link, a GPRS (General Packet Radio Service) link, a GSM (Global System for Mobile Communication) link, a CDMA (Code Division Multiple Access) or TDMA (Time Division Multiple Access) link such as a cellular phone channel, a GPS (Global Positioning System) link, a Bluetooth radio link, or an IEEE 802.11-based radio frequency link. Network 142 may yet further be or include as a segment any one or more of an RS-232 serial connection, IEEE-1394 (Firewire) connections, an IrDA (infrared) port, a SCSI (Small Computer Serial Interface) connection, a USB (Universal Serial Bus) connection or other wired or wireless, digital or analog interfaces or connections.

The network data stream 144 traversing the network 142 in the illustrative embodiment is a sequence of digital bits, which network observation

port 104 senses and collects. Network observation port 104 may be implemented in a computer workstation configured with a network interface card (NIC), with that device configured to promiscuous mode so that all data is communicated transparently through the network observation port 104.

However, in the implementation of the invention, network observation port 104 is preferably embedded in the network without a separate network address, so that its presence on the network is not discernible to network users. Network observation port 104 is likewise preferably installed on a network node, such as a computer workstation or server, which is not responsible for and does not run the network operating system for the network 142. The computer workstation or server which hosts network observation port 104 may be, for instance, a workstation running the Microsoft Windows™ NT™, Unix, Linux, Xenix, Solaris™, OS/2™, BeOS™, Mach, OpenStep™ or other operating system or platform software.

As the realtime network data stream 144 is sensed and collected, the network observation port 104 transmits a copy of the network data stream 144 in the form of collected data stream 106 to interpreter module 108 over connection 146. Interpreter module 108 accepts the collected data stream 106 and interprets the collected data stream 106 into logical groupings, as illustrated in Fig. 4. This process is sometimes called fragment reassembly.

For instance, interpreter module 108 may interpret collected data stream 106 into Ethernet packets in an Ethernet implementation, and strip information

off from those packets that will be extraneous to the further treatment of the collected data stream 106.

In an Ethernet environment, address information in the header reflects a media access control (MAC) hardware address, which is an absolute value and not readily mapped to a user or host, which have a logical rather than physical address. The interpreter module 108 thus removes the portions of the collected data stream 106 which contain the hardware-bound Ethernet header and processes the IP packet content. Interpreter module 108 transmits the resulting data packets 110 over communications link 148 to an assembler module 112.

The assembler module 112 accepts the incoming data packets 110 to perform a next level of data analysis. More particularly, the assembler module 112 consolidates the arriving data packets 110 into complete session files 118 representing discrete network events, such as data access and downloads by individual users. Individual session files 118 may be, for instance, transfer control protocol (TCP) sessions reflecting Internet activity.

As another variety of detectable transmissions, streaming video connections may be transmitted using the user datagram protocol (UDP) standard which is a connectionless protocol, since individual packets do not relate to or depend on preceding or following packets. Given that a UDP packet arrives in data packets 110 and is unique, that packet is added to a reassembly queue 180 (illustrated in Fig. 1) by assembler module 112.

If a subsequent UDP packet arrives with the same IP addresses and the same application ports, before the original packet is marked complete, it will be assumed to be part of the original packet session and reassembled. The criteria for a session to be marked complete in the case of UDP is that the user defined timeout period (preferably with a default such as 30 seconds) is reached, and that the assembler module 112 activates an iterator module 178 on the session. The iterator module 178 only acts when the assembler module 112 enters an idle state, and flushes completed sessions.

Assembler module 112, however, may deduce that a series of data packets 110 containing the same source and destination addresses and traversing the network 142 at the same time are part of a single UDP session, and output a UDP object into session file 118 accordingly. Other protocols may be deduced from the data packets presented to assembler module 112. The assembler module 112 of the invention, for instance, is not limited to recognizing, and does not presume that, all of data packets 110 are arriving under the TCP/IP protocol.

Assembler module 112 may also contain external application port 114 for accepting network packet information collected from separate external applications 116, such as conventional sniffer packages or others.

After storing the sessions into session file 118, the assembler module 112 transmits the sessions 140 to parser module 120 via connection 158.

The parser module 120 stores an overall log of the sessions 140 into session database 122. Parser module 120 contains application sensor module 126 that is invoked for each session 140 to determine the type of application that generated the session. Application sensor 126 uses port assignments, lexical information and other data related to sessions 140 to determine what type of extractor 128 to invoke to process given session 140 Application sensor 126 includes a library of classes of extractors 128 to call up to process sessions 140.

Application sensor 126 characterizes the application type of sessions140 by analyzing a variety of information contained in and characterizing the session 140. That information may include source and destination addresses, sequence numbers, source and destination ports, and other parameters as illustrated in Fig. 5.

Sessions 140 of TCP and other protocols are characterized based in part upon a keyword lexicon analysis. In this regard, parser module 120 contains a lexicon module 174 which analyzes sessions 140 to flag the presence of keyword phrases consistent with different types of TCP sessions. Accumulated information concerning these flags, such as the presence of discreet keywords or totals for keyword occurrences, are used to identify enumerated network objects.

For some types of network information, the occurrence of a single keyword may indicate the presence of an associated data object. For others, the total number of keyword occurrences, a weighted metric or other information

may be compared to a threshold or other criteria to establish that category of event.

For instance, the presence of the phrase "/r/nfrom:" is illustratively flagged for candidacy as both an email and news article object. However, the keyword "/r/nNewsGroup:" correlates only to a news object. The logical trigger for news articles may be the presence of a flag for "/r/nnewsgroup:" being present and flagged. Similarly, the logical trigger for the presence of email may be positive flags for the terms "/r/nFrom:" in addition to the phrase "/r/nTo:".

An example of a procedure call, invoked by the sensor module 126, to identify an SMTP event follows. The code in the following table (illustratively in C++, although it will be understood that other languages may be used) may be employed according to the invention to isolate those types of mail transmissions.

Table 1

```
^HELO {
                FlagIt (APP_STATE, APP_SMTP, SMTPHELO);
        }
^data[$] {
                FlagIt (APP_STATE, APP_SMTP, SMTPDATA);
        }
^data\r {
                FlagIt (APP_STATE, APP_SMTP, SMTPDATA);
        }
^"mail from"[ ]*: {
                FlagIt (APP_STATE, APP_SMTP, SMTPMAILFROM);
        }
^"rcpt to"[ ]*: {
                FlagIt (APP_STATE, APP_SMTP, SMTPRCPTTO);
        }
```

```
^EHLO   {
                FlagIt (APP_STATE, APP_SMTP, SMTPHELO);
        }

#define MINSMTPMATCH(X) ((X) & SMTPHELO && (X) & SMTPDATA
&& (X) & SMTPRCPTTO)
```

According to the foregoing procedure call, each occurrence of the word "HELO" preceded by a line feed ('^') is flagged as a SMTPHELO. According to the Minimum Match Criteria (MINSMTPMATCH), if a 'SMTPHELO', 'SMTPDATA', and 'SMTPRCPTTO' is found, the match is made and an SMTP parser is called.

Similarly, in terms of profiling and triggering a HTTP/HTML event, the following procedure call may be employed.

Table 2

```
"GET " {  /*BEGINNING of HTTP STUFF */
                FlagIt (APP_STATE, APP_HTTP, HTTPGET);
  }
"Referer: " {
                FlagIt (APP_STATE, APP_HTTP, HTTPREFERER);
  }
"Accept: " {
                FlagIt (APP_STATE, APP_HTTP, HTTPACCEPT);
  }
"User-Agent: " {
                FlagIt (APP_STATE, APP_HTTP, HTTPUSERAGENT);
  }

"HTTP/"[0-9]"."[0-9] {
                FlagIt (APP_STATE, APP_HTTP, HTTPVERSION);
        }

 /*  HTML FLAGS  */
```

```
"<HTML" {
                FlagIt (CONTENT_STATE, CNT_HTML, HTMLTAG);
  }
"<A HREF" {
                FlagIt (CONTENT_STATE, CNT_HTML, HTMLHREF);
  }
"<H1" {
                FlagIt (CONTENT_STATE, CNT_HTML, HTMLH1);
  }
"</a" {
                FlagIt (CONTENT_STATE, CNT_HTML, HTMLANCHOR);
  }

"<HEAD>" {
                FlagIt (CONTENT_STATE, CNT_HTML, HTMLHEAD);
                }

"<BODY" {
                FlagIt (CONTENT_STATE, CNT_HTML, HTMLBODY);
                }

#define MINHTTPMATCH(X) ((X) & HTTPVERSION)
#define MINHTMLMATCH(X) ((X) & HTMLTAG && (X) & HTMLHEAD
&& (X) & HTMLBODY)
```

Other protocols may be triggered upon other corresponding lexical triggers, or other types of information when the network event is not textually-based. For example, the original network data stream 144 may be sampled during a streaming video, voice-over-network or other virtual connections which are not encapsulated in a textual or TCP format.

Because network protocols may be nested, for example, a POP-3 session may contain one or more instances of RFC822 email sessions, application sensor 126 may be applied recursively to identify protocols within other

protocols to extract nested or underlying objects encapsulated in one or more different protocols.

The protocols the invention may detect include, but are not limited to, TCP, IP, UDP, SMTP, HTTP, NNTP, FTP, TELNET, DNS, RIP, BGP, MAIL, NEWS, HTML, XML, PGP, S/MIME, POP, IMAP, V-CARD, ICMP, NetBUI, IPX and SPX objects, understood by persons skilled in the art. The universe of protocols that sensor module 128 can detect and identify is extensible, and can be added to or subtracted from to accommodate future protocols and for other network needs.

Once application type of session 140 has been determined by application sensor 126, parser module 120 may, depending upon configuration information and type of session, store part or all of a complete session to content database 182 after assignation of a unique storage address.

The parser module 120 also contains extractor module 128, which processes the determined protocol for a given session 140 and generates the minimum subset of information needed to identify the nature of session 140 for recording on session database 122, removing unnecessary information before storage. Information may be reduced using text compression and other techniques. Because network protocols are designed to nest, extractor 128 is applied recursively to process protocols within other protocols, as identified by sensor 126. Depending on the category of session 140, the data reduction from

the original network sessions to the metadata image of the session (each stored on session database 122) may be on the order of 100 to 1 or greater.

Depending on the size of network 142, the bandwidth of network data stream 144 and other factors, the storage requirements of session database 122 may be substantial. However, the storage requirement of the invention is commensurate with the comprehensive nature of the surveillance performed and affords system administrators the opportunity to perform more fully featured post hoc traffic analysis.

At the back end of the network apparatus of the invention, a presentation interface 138 (illustrated in more detail in Fig. 3) communicates via communication line 168 to a presentation server 136. The presentation server 136 may be a workstation or other device, such as a personal computer running the Microsoft Windows™ 95, 98, NT™, Unix, Linux, Solaris™, OS/2™, BeOS™, MacOS™ or other operating system. The presentation interface 138 may be accessed by a systems administrator wishing to perform network investigation or maintenance, and may be connected to presentation server 136 for example via a common gateway interface (CGI) bin or other Web service interfaces.

The presentation server 136 is in turn connected via communications link 166 to a summary database 132, which is in turn connected via connection 164 to session database 122. The session database 122 and summary database 132 may in one regard be serviced by the same database engine, such as an

online analytic processing (OLAP) interface. Execution of scripts through an OLAP or other engine such as a relational database engine accessed by Standard Query Language (SQL) generates the summary database 132 from searches on the session database 122.

Presentation interface 138 allows a systems administrator to invoke a graphical or other menu of different inquiries into the past behavior of network 142. Those inquiries may include an investigation of Websites most frequently visited by users of the network, individual users exhibiting the highest rate of e-mail traffic including images of the e-mail messages themselves, nodal analyses of different network addresses and their most frequent communicants, and other information recorded in the resulting databases.

The variety of forensic inquires that may be formulated through presentation interface 138 is in part a function of the complete nature of the surveillance performed by the invention, and the storage of the results of those interrogations in summary database 132 also allows further treatment by characterization module 134 communicating with summary database 132 over connection 172.

The characterization module 134 may store high-level, digested data indicating the overall behavior of network 142, such as peak traffic times, distribution of utilized bandwidths across the network over time, general degree of user activity and other categories of characteristic data.

Presentation interface 138 may overlay the graphical or other depiction of the network behavior with system policy constraints or goals, such as limits on Web access or e-mail traffic, to visually show how different facets of the network are complying or behaving. Presentation interface 138 may, if desired, be connected to a printer or other output device (not shown) to produce hard copy of the different varieties of reports prepared according to the invention.

Similarly, summary database 132 may include ports to other external applications to receive further collateral information concerning network behavior, such as employee lists, accounting records and other packages.

The overall processing flow of the invention is illustrated in Figure 2. In step 202, processing begins. In step 204, bits from the network data stream 144 are collected by network observation port 104 into collected data stream 106. In step 206, the collected data stream 106 is transmitted to interpreter module 108. In step 208, the interpreter module 108 resolves the collected data stream 106 into data packets 110. In step 210, the assembler module 112 accepts additional packets from any external application ports, if any are present.

In step 212, assembler module 112 assembles data packets 110 into individual sessions 140, storing new sessions in session file 118. In step 214, assembler module 112 transmits copies of the sessions 140 to parser module 120. In step 216, the parser module 120 invokes the sensor module 126 to assign a session type to individual sessions 140.

In step 218, the extractor module 128 is invoked to extract the minimum essential session data to be reflected in summary database 132. In step 220, parsed session information is stored in session database 122. In step 222, the summary database 132 is generated by executing OLAP scripts or other search or query mechanisms against session database 122. In step 224, the presentation interface 138 is presented to a systems administrator or other user.

In step 226, a user inquiry is accepted, such as an interrogation from a systems administrator. In step 228, the user inquiry is input to the presentation server 136. In step 230, the presentation server 136 analyzes the query parameters and communicates with the summary database 132. In step 232, the characterization module 134 is executed. In step 234, the resulting graphical or other data are presented to the user via the presentation interface 138. In step 236, processing ends.

The foregoing description of the system and method of the invention is illustrative, and variations in configurations and implementation will be apparent to persons skilled in the art. For instance, while the interpreter module 108 has been illustrated as accepting input form a single network observation port 104, interpreter module 108 could accept samples of the network data stream 144 from multiple ports.

Similarly, while presentation interface 138 has been illustrated as an interactive module accepting analytic requests from a user, predetermined sets of reports can be executed by presentation server 136, summary database 132

and associated components in batch fashion. While certain functions have been described as being stored on and executed by individual modules, servers and other network elements, it will be appreciated that different aspects of the control and analysis of the invention maybe executed by different computers or other devices, in distributed fashion. The scope of the invention is accordingly intended only to be limited by the following claims.